

CART & MARS

Methodology Training Document (Module 6)

YEAR 2015



Objectives and Scope

Course Goals

- Introduction to importance of using CART and MARS in line with EXL DA Methodology
- Provide a structured overview of every tab in CART and MARS model set up
- Explain output interpretation
- Give hands-on training on CART, subject to availability of server access and CART license
- Provide helpful “tricks of the trade”

Beyond the Scope of this Training

- Comprehensive coaching on every feature of CART and MARS software
- Backend algorithms of CART and MARS software

Self Study Goals

- Exploration of various features of CART and MARS by practising on some dummy data
- Innovations and new features of CART and MARS as and when released by Salford Systems
- Discussion on advanced features can be taken up offline

Table of Contents

- I. Introduction to CART**
- II. CART Basics**
- III. Setting Up CART Model**
- IV. CART Output**
- V. Introduction to MARS**
- VI. Spline Transformation**
- VII. Setting Up MARS Model**
- VIII. MARS Output**
- IX. CART and MARS for Creating Transforms**

I. Introduction to CART

CLASSIFICATION AND REGRESSION TREES (CART)

- CART is a robust decision-tree tool for data mining, predictive modeling, and data preprocessing.
- CART automatically searches for important patterns and relationships, uncovering hidden structure even in highly complex data.
- CART trees can be used to generate accurate and reliable predictive models for a broad range of applications from bioinformatics to risk management.
- The most common applications include churn prediction, credit scoring, drug discovery, fraud detection, manufacturing quality control, and wildlife research.

Note: Several hundred detailed applications studies are available at <http://www.salford-systems.com>

II. CART Basics

Points to Remember:

- Maximum number of cells (rows x columns) allowed in the analysis is **limited by license**
- CART is **case insensitive** for variable names; all reports show variables in upper case
- CART supports both character and numeric variable values
- Variable names **must not exceed** 32 characters
- Variable names must have only **letters, numbers** or **underscores**
- Variable names must start with a letter
- When a variable name ends with “\$” and/or if the data value is surrounded by quotes on the first record, it is processed as a character variable. In this case, a “\$” sign is added to the variable name if needed.

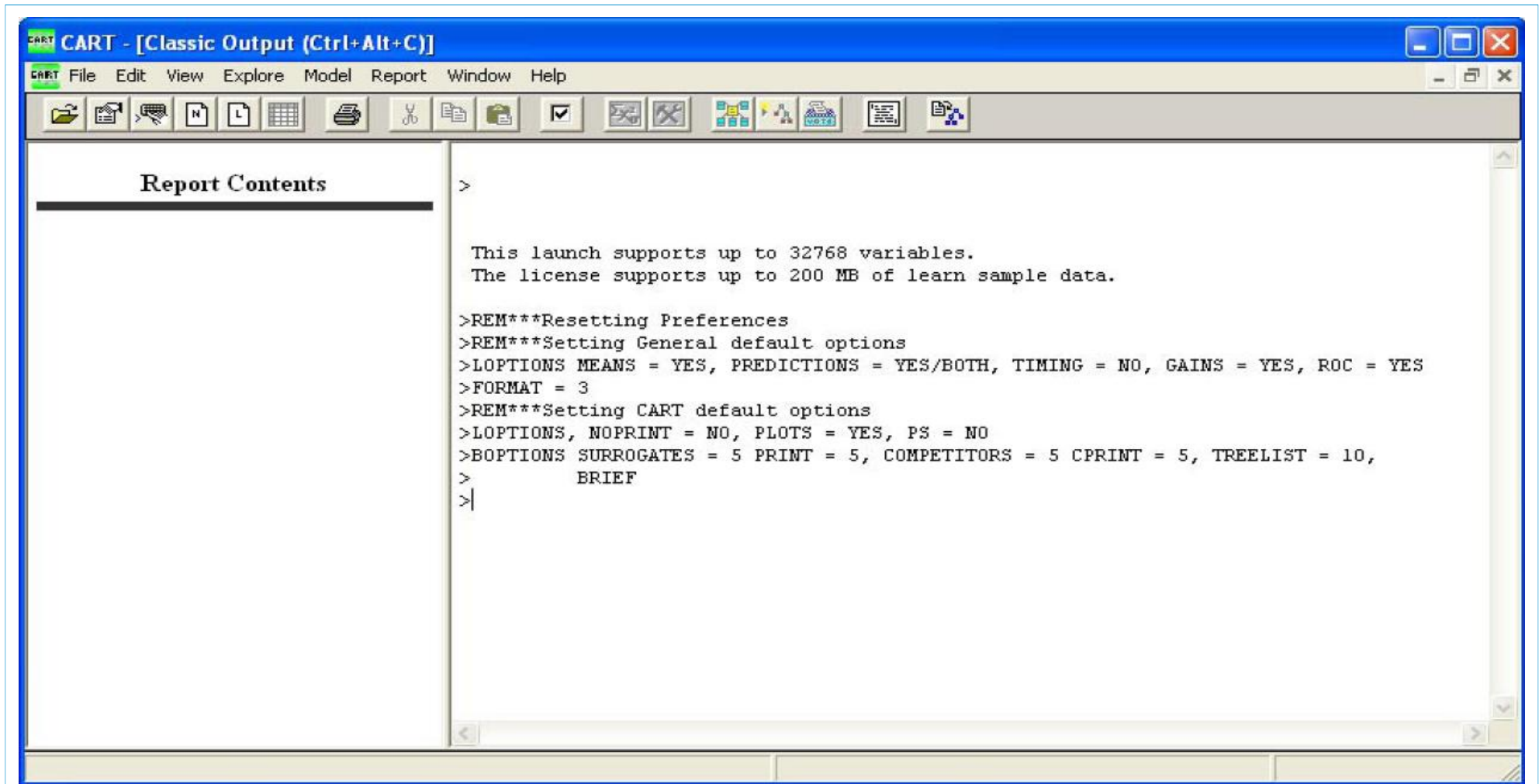
Examples on Acceptable/Unacceptable Variable Names:

VARIABLE NAME	TYPE	COMMENT
AGE	Numeric	OK
AGE_1	Numeric	OK
1PAYMENT	Numeric	Unacceptable; leading character other than letter.
%WEIGHT	Numeric	Unacceptable; leading character other than letter.
SOCIAL_SECURITY_NUMBER_AND_ACCOUNT	Numeric	Unacceptable; too long. Variable name will be truncated to 32 characters.
DEPOSIT&AMOUNT	Numeric	Unacceptable; “&” is not letter, number or underscore. This character will be replaced with an underscore.
NAME	Character	OK. But being a character variable, “\$” will automatically be added at the end. This variable name would be displayed as “NAME\$”.

II. CART Basics

CART Desktop:

This is the first screen that pops up as we double-click on CART program icon.



II. CART Basics

An Overview Layout of Main CART MENU Options:

- **FILE**
 - Open Dataset, Navigator, Grove or Command File
 - Save Analysis Results, Navigator, Grove or Command File
 - Open a CART notepad for creating command scripts
 - Specify printing parameters
 - Activate interactive command mode
 - Submit batch command files
- **EDIT**
 - Cut, Copy and Paste selected text
 - Specify colors and fonts
 - Control reporting options
 - Specify default directories
- **VIEW**
 - Open command log
 - View Data
 - View descriptive statistics
 - Display next pruning
 - Assign class names and apply colors
 - View main tree and/or sub-tree rules
 - Overlay gains chart
 - Specify level of detail displayed in tree nodes
- **EXPLORE**
 - Generate Frequency Distributions
- **MODEL**
 - Specify model set-up parameters
 - Grow trees/committee of experts
 - Generate predictions/score data
 - Translate models into SAS, C or PMML
- **REPORT**
 - Control CART reporting facility
- **WINDOW**
 - Control various windows on the CART desktop
- **HELP**
 - Access online help

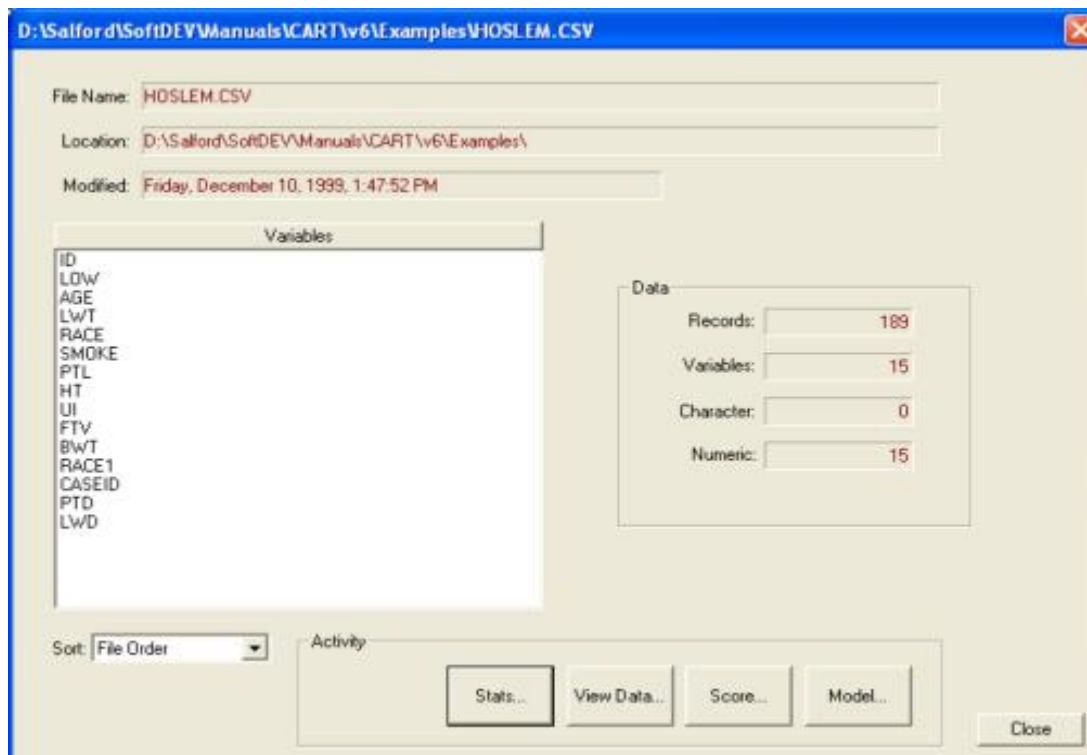
II. CART Basics

Opening a File:

- Select Open → Data File ... from the **File** menu (or click on the  toolbar icon)
- Navigate to the file location and select the file to open.

Note: As an alternative to navigation, default input and output directories can be reset; select Options ... from the **Edit** menu and select the **Directories** tab

Activity Dialog Box:



Example here imports a CSV file. The data file, however, could be a SAS dataset.

The file contains 189 records and 15 variables and all are numeric variables.

Using Sort drop-down control, variables can be sorted in either Alphabetical or File Order.

III. Setting Up CART Model

THE MODEL TAB

- Tab heading is displayed in **RED** if tab *requires information from user* before a model can be built.
- **Target Variable Selection**
 - The moment a **TARGET** variable is selected, CART knows which one of all the variables is to be analyzed or predicted. This is *the only required step* in setting up a model. Everything else is optional.
- **Tree Type**
 - **Classification Tree**: It uses a “*categorical*” target variable (e.g.; YES/NO). The purpose of classification is to accurately discriminate between classes.
 - **Regression Tree**: it uses a “*continuous*” variable (such as AGE or INCOME). The purpose of regression is to predict values that are close to a true outcome.
- **Predictor Variable Selection**
 - Candidate predictor (independent) variables are specified by check marks in the Predictor column
- **Case Weights**
 - To select a variable as the case weight, simply put a checkmark against that variable in the **Weight** column.
 - An observation’s case weight can be thought of as a repetition factor. It may take on fractional values or whole numbers.
 - A missing, negative or zero case weight causes the observation to be deleted, just as if the target variable were missing.

III. Setting Up CART Model

THE MODEL TAB

- Auxiliary Variables**
 - Auxiliary variables are variables that are tracked throughout the CART tree but are not necessarily used as predictors.
 - By marking a variable as Auxiliary, it is indicated that basic summary statistics for such a variable may be retrieved later for any node in the CART tree.
- Setting Focus Class**
 - In classification runs, some of the reports generated by CART (gains, prediction success, color coding etc.) have one target class in focus.

By default, CART will put the first class it finds in the dataset *in focus*.

A Snapshot of Model Tab

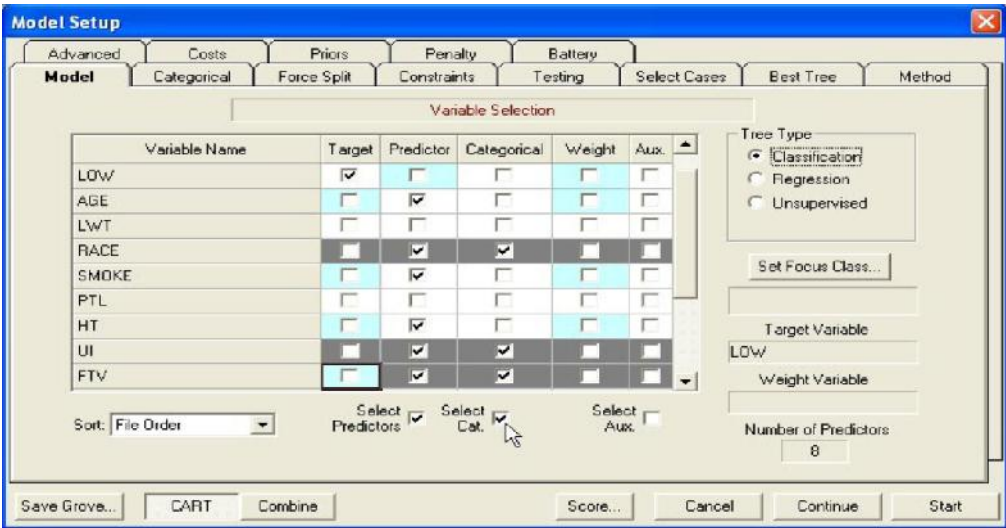
The binary categorical variable “LOW” (coded 0/1) is the target (or dependent) variable.

For categorical dependent variable, Tree Type ought to be selected as “Classification”.

8 variables have been selected as predictor variables.

Using Sort drop-down control, variables can be sorted in either Alphabetical or File Order.

Note: Each of the Model Setup tabs contains a [Save Grove...] button in the lower left corner. It helps saving the model for future review, scoring or export.



III. Setting Up CART Model

THE CATEGORICAL TAB

- **Setting Class Names**
 - Select variable. Press [Set Class Names] option to enter/edit class names for the variable.
- **High Level Categorical (HLC) Predictors**
 - Such variables present computational challenge because of exploding number of possible ways to split the data in a node.

#Levels	#Distinct Splits
K	$2^{K-1} - 1$
2	1
3	3
4	7
⋮	⋮
⋮	⋮
⋮	⋮
10	511
⋮	⋮
⋮	⋮
⋮	⋮
21	Over a million !!
⋮	⋮
⋮	⋮
⋮	⋮

For binary target variable, CART has special shortcuts for HLC predictors that always work. HLC settings, thus, are relevant only if target variable has more than 2 levels.

The threshold level of 15 indicates that

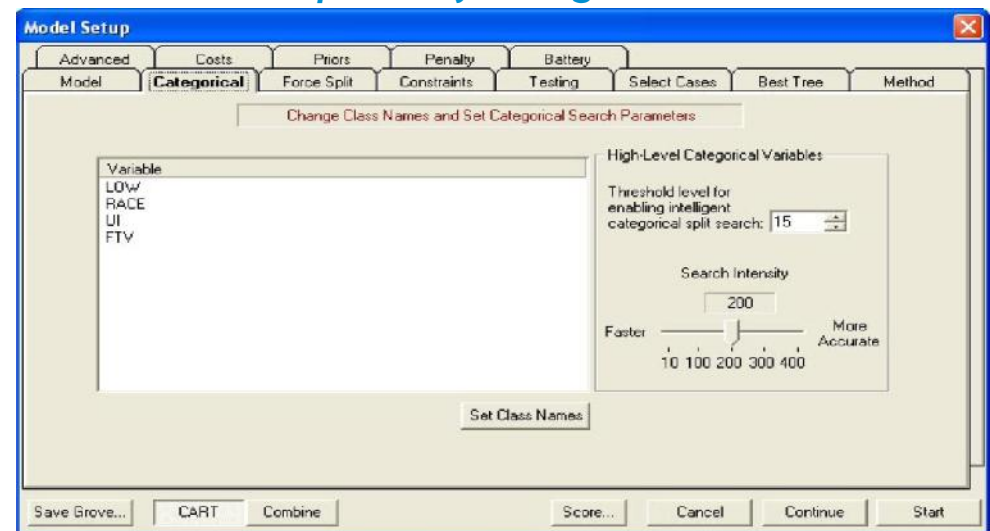
- for categorical predictors with 15 or fewer levels, CART would search all possible splits and definitely find the overall best partition
- for predictors with more than 15 levels, intelligent shortcuts would be used to do fast “local” searches for very good partitions (though these may not be the absolute overall best).

As the short-cut method explore only a limited range of possible splits, *search intensity can be set within a range of 0-400*. The default setting is 200.



Searching too aggressively for the best HLC splitter increases the likelihood of over-fitting the model to the training data.

A Snapshot of Categorical Tab



III. Setting Up CART Model

THE TESTING TAB

- **No independent testing - exploratory tree**
 - This option *skips the entire testing phase* and simply reports the largest tree grown.
 - Because no test method is specified, CART does not select an “optimal” tree.
 - Bypassing the test phase can be useful when CART is being used to generate a quick cross tabulation of the target against one of the predictors. It is also useful for “supervised binning” or aggregation of variables such as high-level categorical candidates.
- **Fraction of cases selected at random for testing**
 - Use this option to let CART *automatically separate a specified percentage of data for test purposes*. Because no optimal fraction is best for all situations, user may want to experiment.
 - This mechanism does not provide user with a way of tagging the records used for testing.
- **Variable separate learn, test, (validate)**
 - A variable on the data set can be used to flag which records are to be used for learning (training) and which are to be used for testing or validation.
 - Use a *binary (0/1) numeric variable* to define simple *learn/test partitions*.
- **Test sample contained in a separate file**
 - *Two separate files* are assumed — one for *learning* and one for *testing*. The files can be in different database formats and their columns do not need to be in the same order.
 - The train and test files must both contain ALL variables to be used in the modeling process.

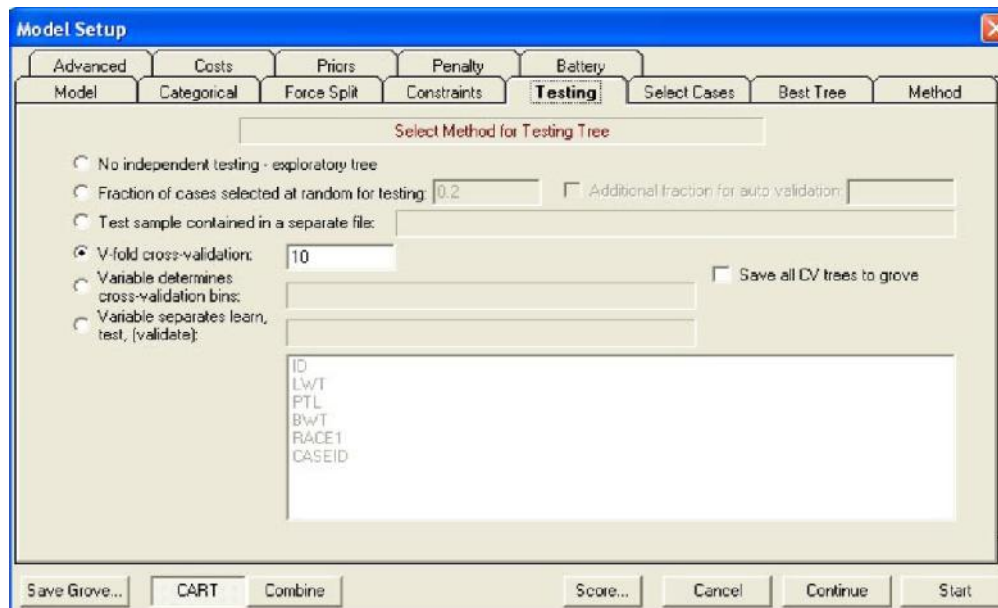
III. Setting Up CART Model

THE TESTING TAB

■ V-fold cross-validation

- Cross validation is a *marvelous way to make the maximal use of training data*, although it is typically used when data sets are small.
- Even if dataset is large, in case of low event rate, cross-validation may be the only viable testing method.
- Cross validation allows user to build tree using all the data. The testing phase requires running an additional ‘V’ trees (in V-fold CV), each of which is tested on a different V% of the data. The results from those ‘V’ test runs are combined to create a table of synthesized test results.

A Snapshot of Testing Tab



Default test setting: 10-fold cross validation

*Note: Every target class must have at least as many records as the number of folds in the cross validation. Otherwise, the process breaks down, an **error message** is reported, and a “No Tree Built” situation occurs. This means that if data set contains only nine YES records in a YES/NO problem, CART cannot run more than nine-fold cross validation.*

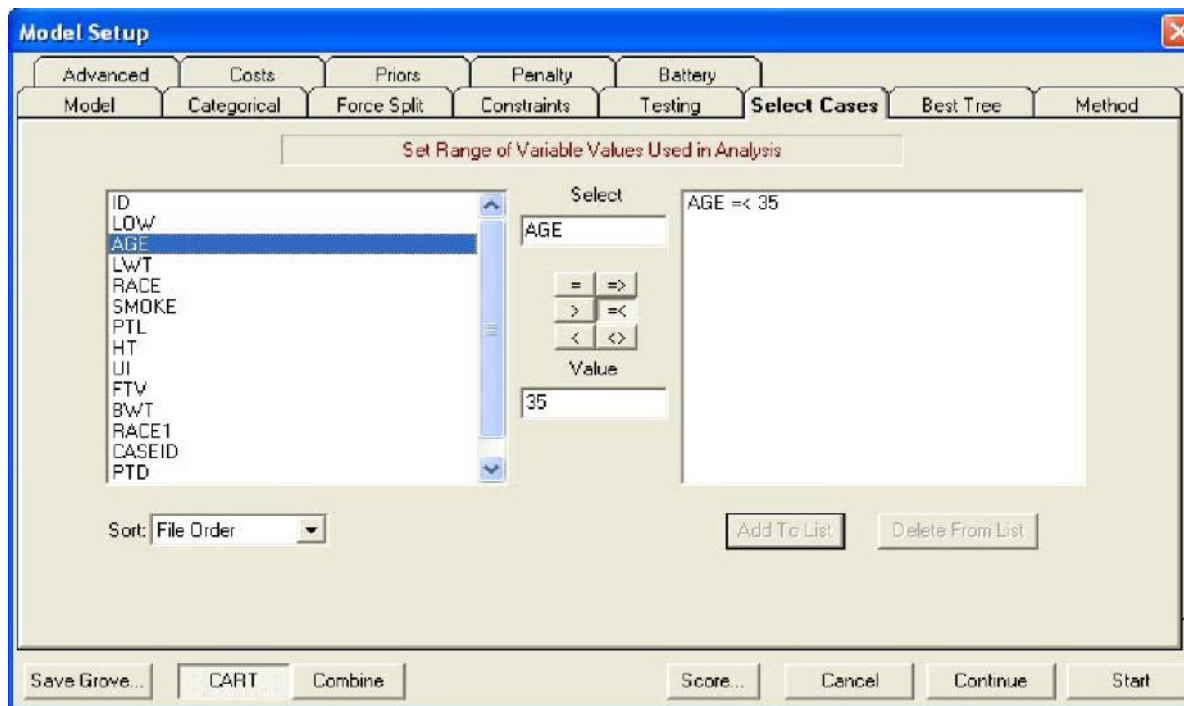
III. Setting Up CART Model

THE SELECT CASES TAB

- The Model Setup – Select Cases tab allows the user to specify **up to ten selection criteria** for building a tree based on a subset of cases.

A selection criterion can be specified in terms of *any variable appearing in the data set*, whether or not that variable is involved in the model.

A Snapshot of Select Cases Tab



Steps

1. Double-click a variable in the variable list to add that variable to the **Select** text box.
2. Select one of the predefined logical relations by clicking its radio button.
3. Enter a numerical value in the **Value** text box.
4. Click **[Add to List]** to add the constructed criterion to the right window and use **[Delete from List]** to remove.

III. Setting Up CART Model

THE BEST TREE TAB

■ Default Best Tree settings:

- *Minimum cost tree regardless of size* (most accurate tree, given the specified testing method)

Note: Alternatively, the user may wish to trade a more accurate tree for a smaller tree by selecting the smallest tree within one standard error of the minimum cost tree or by setting the standard error parameter equal to any nonnegative value.

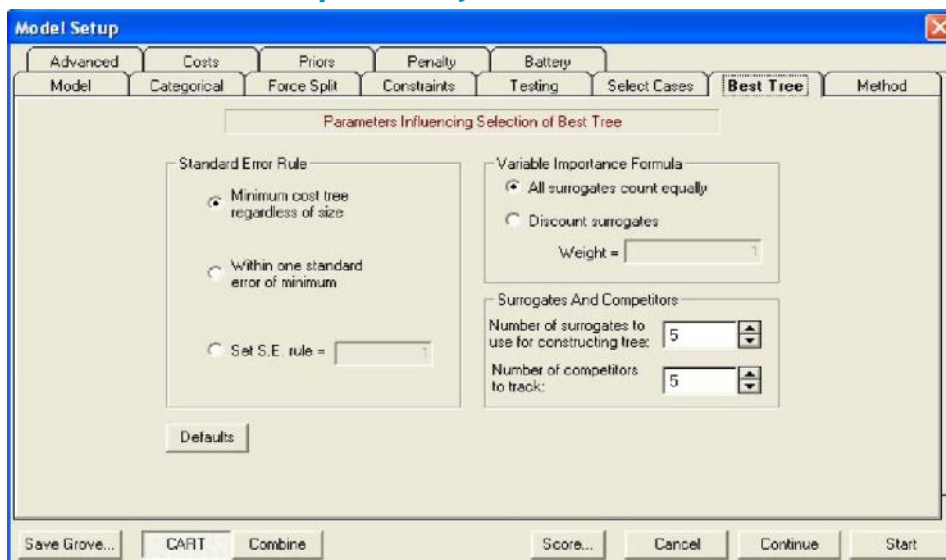
- *Five surrogates* used to construct tree

Note: Alternatively, the user can increase or decrease the number of surrogates that CART searches for.

- *All surrogates count equally* to compute variable importance

Note: Alternatively, the user can fine-tune the variable importance calculation by specifying a weight to be used to discount the surrogates. Click on the Discount surrogates radio button and enter a value between 0 and 1 in the Weight text box.

A Snapshot of Best Tree Tab



Note: Surrogates refer to *splitters* that are similar to the primary splitter and can be used when the primary split variable is missing.

The Model Setup–Best Tree tab is largely of *historical interest* as it dates to a time when CART would produce a single tree in any run.

In today's CART, user has full access to every tree in the pruned tree sequence and can readily select trees of a size different than considered optimal.

III. Setting Up CART Model

THE METHOD TAB

■ Classification Tree Splitting Rules:

- **Gini:** This default rule *often works well across a broad range of problems*. Gini has a tendency to generate trees that include some rather small nodes highly concentrated with the class of interest.
- **Symmetric Gini:** This is a special variant of the Gini rule designed specifically to work with a cost matrix. If different costs for different classification errors are not specified, the Gini and the Symmetric Gini are identical.
- **Entropy:** This tends to produce even smaller terminal nodes (“end cut splits”) and is usually *less accurate than Gini*.
- **Class Probability:** Probability trees tend to be larger than Gini trees and the predictions made in individual terminal nodes tend to be less reliable, but the details of the data structure that they reveal can be very valuable. When the user is primarily interested in *performance of top few nodes of a tree, probability trees are more useful*.
- **Twoing:** The major difference between the Twoing and other splitting rules is that *Twoing tends to produce more balanced splits (in size)*. Twoing has a built-in penalty that makes it avoid unequal splits. A Gini or Entropy tree could easily produce 90/10 splits whereas Twoing will tend to produce 50/50 splits. The differences between the Twoing and other rules become more evident in case of multi-class target variable.
- **Ordered Twoing:** The Ordered Twoing rule is *useful when target levels are ordered classes*. Ordered Twoing can be thought of as developing a model that is somewhere between a classification and a regression. Remember that the other splitting rules would not care at all which levels were grouped together because they ignore the numeric significance of the class label.

III. Setting Up CART Model

THE METHOD TAB

■ Regression Tree Splitting Methods:

- *Least Squares*: The objective function is to minimize sum of square of errors.
- *Least Absolute Deviation*: The objective function is to minimize sum of absolute errors.

In general, Least Squares Method is preferred, as the error magnitude itself is assigned as ‘weight’ to the corresponding error term.

■ Favor Even Splits:

- By default, the setting is 0, which indicates no bias in favor of even or uneven splits.

On binary targets when both “Favor even splits” and unit cost matrix is set to 0, Gini, Symmetric Gini, Twoing, and Ordered Twoing will produce near identical results.

■ Linear Combination Splits:

- To deal more effectively with linear structure, CART has an option that allows node splits to be made on linear combinations of non-categorical variables. This option is implemented by clicking on the **Use Linear Combinations for Splitting** check box on the Method tab.



CART will terminate the model-building process prematurely if it finds that it needs more linear combination splits than were actually reserved.

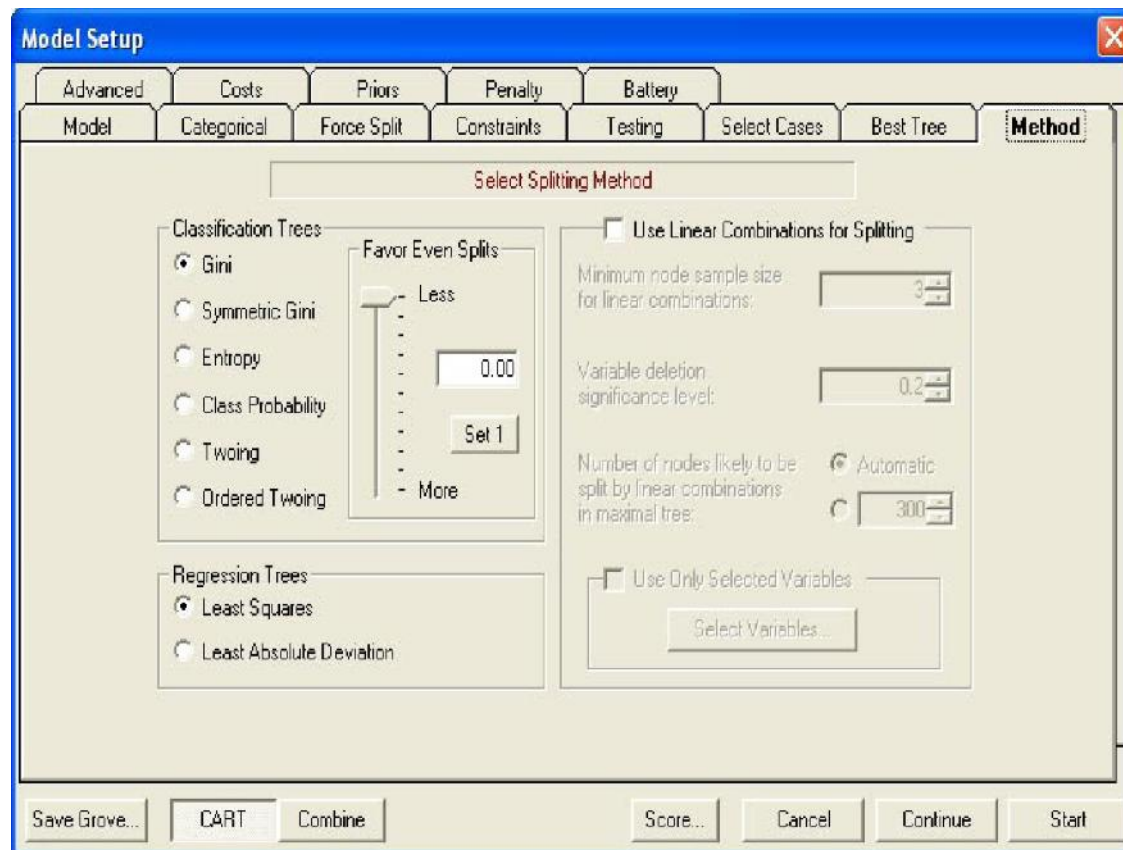


Linear combination splits will be automatically turned off for all nodes that have any constant predictors (all values the same for all records). Thus, having a constant predictor in the training data will effectively turn off linear combinations for the entire tree.

III. Setting Up CART Model

THE METHOD TAB

A Snapshot of Method Tab



For a *two-level* dependent variable that can be predicted with a *relative error of less than 0.50*, the *Gini splitting rule* is typically best.

For a *two-level* dependent variable that can be predicted with a *relative error of only 0.80 or higher*, *Power-Modified Twoing* tends to perform best.

For target variables with *four to nine levels*, *Twoing* has a good chance of being the best splitting rule.

For higher-level categorical dependent variables with *10 or more levels*, *Twoing* or *Power-Modified Twoing* is often considerably more accurate than Gini.

III. Setting Up CART Model

THE ADVANCED TAB

■ Minimum Node Size:

- *Parent Node Minimum Cases:* In search of best optimal tree, CART may continue splitting nodes till it runs out of data. However, nodes with negligible size are practically irrelevant. This option allows the user to set a minimum number of records for all parent nodes of the tree.
- *Terminal Node Minimum Cases:* This control specifies the smallest number of observations that may be separated into a child node.

As a rule of thumb, parent node's minimum size should be at least three times the minimum size of terminal node.

■ Minimum Complexity:

- Setting complexity to a value greater than the default value 'zero' places a penalty on larger trees, and causes CART to stop its tree-growing process before reaching the largest possible tree size a child node.

■ Tree Size:

- Used to specify 'Maximum number of nodes' (internal plus terminal) and 'Depth' (the root node corresponds to the depth of zero!)

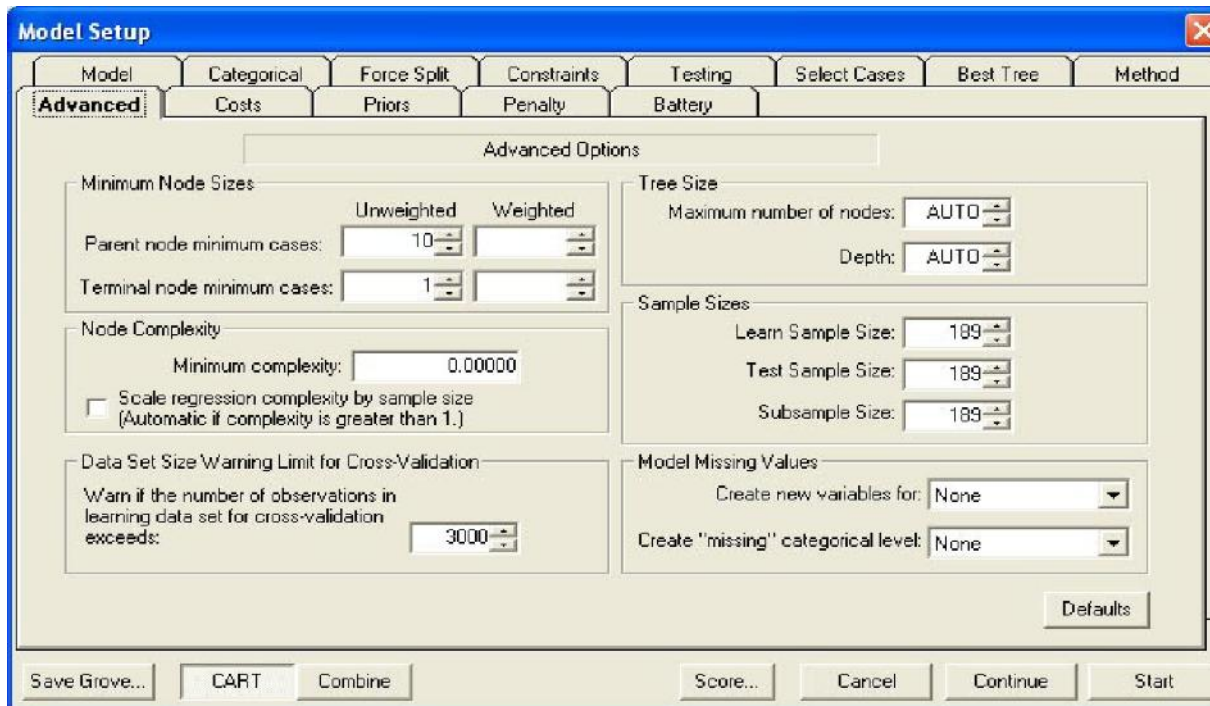
■ Sample Size:

- *Learn Sample Size:* This option limits CART to processing only the first part of the data available and simply ignoring any data that comes after the allowed records. The control allows for faster processing of the data because the entire data file is never read.
- *Test Sample Size:* The TEST setting is similar to LEARN.
- *Subsample Size:* In node sub-sampling, the tree generation process continues to work with the complete data set in all respects except for the split search procedure (which is conducted on a specified size of node sub-sample).

III. Setting Up CART Model

THE ADVANCED TAB

A Snapshot of Advanced Tab



Model Setup

Model | Categorical | Force Split | Constraints | Testing | Select Cases | Best Tree | Method

Advanced | Costs | Priors | Penalty | Battery

Advanced Options

Minimum Node Sizes:

Unweighted | Weighted

Parent node minimum cases: 10 |

Terminal node minimum cases: 1 |

Node Complexity:

Minimum complexity: 0.00000

☐ Scale regression complexity by sample size (Automatic if complexity is greater than 1.)

Data Set Size Warning Limit for Cross-Validation:

Warn if the number of observations in learning data set for cross-validation exceeds: 3000

Tree Size:

Maximum number of nodes: AUTO

Depth: AUTO

Sample Sizes:

Learn Sample Size: 189

Test Sample Size: 189

Subsample Size: 189

Model Missing Values:

Create new variables for: None

Create "missing" categorical level: None

Defaults

Save Grove... | CART | Combine | Score... | Cancel | Continue | Start

By default, CART sets the maximum DEPTH value so large that it will never be reached.

Unlike complexity, the **NODES** and **DEPTH** controls may handicap the tree and result in inferior performance.



Usually decision tree users tend to set depth values to small limits such as five or eight. These limits are generally set very low to create the illusion of fast data processing. However, if user wants to be sure to get the best tree, there is a need to allow for somewhat deeper trees.

III. Setting Up CART Model

THE COSTS TAB

TP / TN / FP / FN CONCEPT

Actual Class	Predicted Class	Prediction Type	Match Result	Classification Type
1	1	POSITIVE	TRUE	TRUE POSITIVE
0	0	NEGATIVE	TRUE	TRUE NEGATIVE
1	0	NEGATIVE	FALSE	FALSE NEGATIVE
0	1	POSITIVE	FALSE	FALSE POSITIVE

 Correct Classification
 Misclassification



Every mistake is not equally serious or equally costly !!

Classic Example: Medical Test

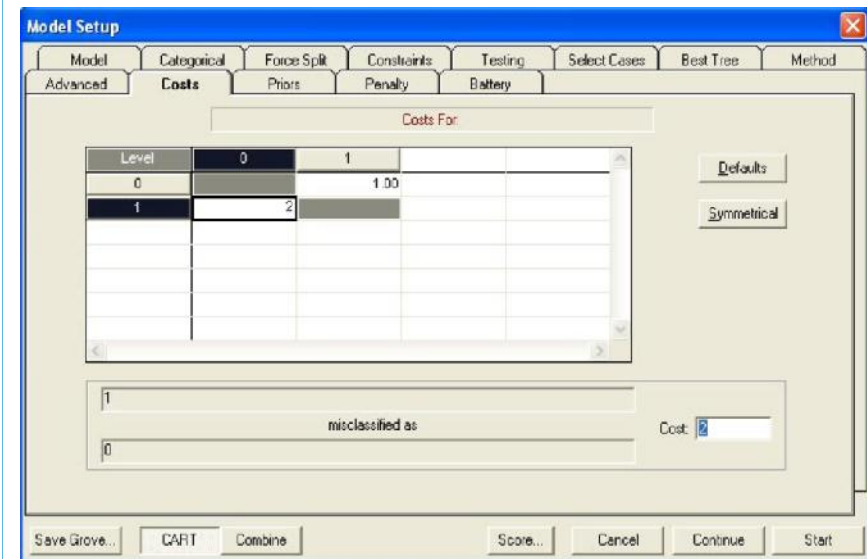
A **false positive** on a medical test might cause additional more costly tests amounting to several hundreds of dollars. A **false negative** might allow a potentially life-threatening illness to go untreated.

Note:

Only cell ratios matter, that is, the actual value in each cell of the cost matrix is of no consequence—setting costs to 1 and 2 for the binary case is equivalent to setting costs to 10 and 20.

CART requires **all costs to be strictly positive** (zero is not allowed). Use small values, such as .001, to effectively impose zero costs in some cells.

A Snapshot of Costs Tab



Model Setup

Model: Categorical Force Split Constraints Testing Select Cases Best Tree Method

Advanced Costs Priors Penalty Battery

Costs For:

Level	0	1
0		1.00
1	2	

Defaults
Symmetrical

1 misclassified as 0 Cost: 1.00

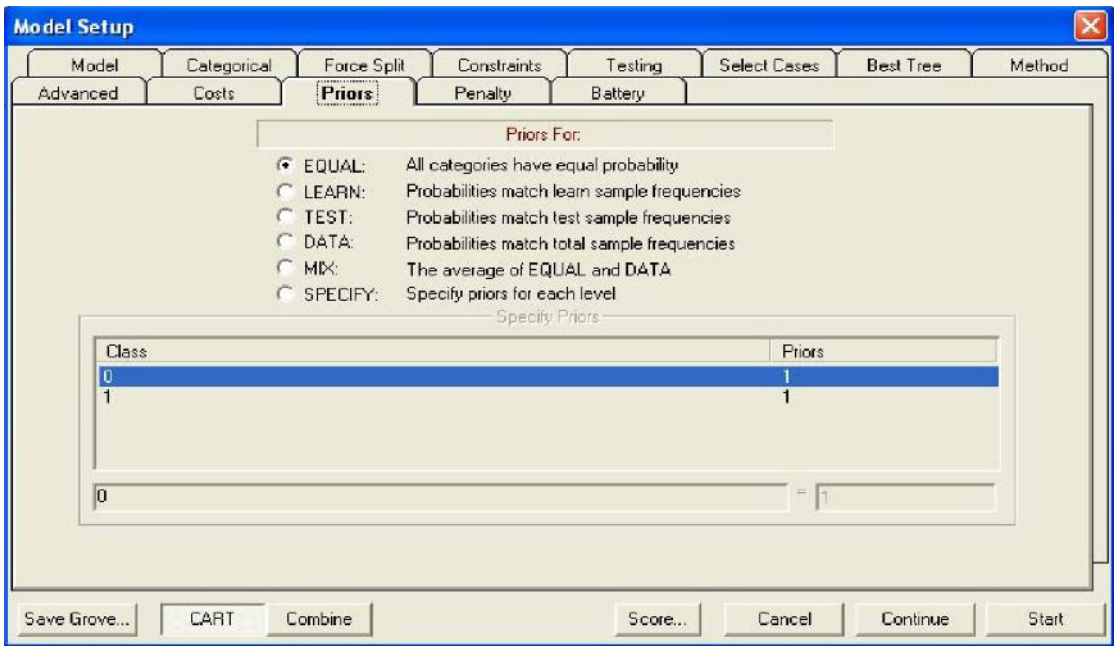
Save Grove... CART Combine Score... Cancel Continue Start

III. Setting Up CART Model

THE PRIORS TAB

- Six different priors options are available, as follows:
- EQUAL** : Equivalent to weighting classes to achieve BALANCE
 - DATA** : Larger classes are allowed to dominate the analysis
 - MIX** : Priors set to the average of the DATA and EQUAL options
 - LEARN** : Class sizes calculated from LEARN sample only
 - TEST** : Class sizes calculated from TEST sample only
 - SPECIFY** : Priors set to user-specified values

A Snapshot of Priors Tab



Model Setup

Model Categorical Force Split Constraints Testing Select Cases Best Tree Method

Advanced Costs **Priors** Penalty Battery

Priors For:

- ☒ **EQUAL:** All categories have equal probability
- ☐ **LEARN:** Probabilities match learn sample frequencies
- ☐ **TEST:** Probabilities match test sample frequencies
- ☐ **DATA:** Probabilities match total sample frequencies
- ☐ **MIX:** The average of EQUAL and DATA
- ☐ **SPECIFY:** Specify priors for each level

Specify Priors

Class	Priors
0	1
1	1

0 = 1

Save Grove... CART Combine Score... Cancel Continue Start

EQUAL PRIORS

It is the default method used for classification trees to deal with unbalanced data (event rate $\neq 0.50$) and **often works supremely well**. Each class is treated as equally important for the purpose of achieving classification accuracy.

Priors are usually specified as fractions that sum to 1.0. In a two-class problem EQUAL priors would be expressed numerically as 0.50, 0.50, and in a three-class problem they would be expressed as 0.333, 0.333, 0.333.

Other Options

PRIORS DATA (or **PRIORS LEARN** or **PRIORS TEST**) makes no adjustments for relative class sizes. Under this setting small classes will have less influence on the CART tree and may even be ignored if they interfere with CART's ability to classify the larger classes accurately. **PRIORS DATA** is perfectly reasonable when the importance of classification accuracy is proportional to class size.

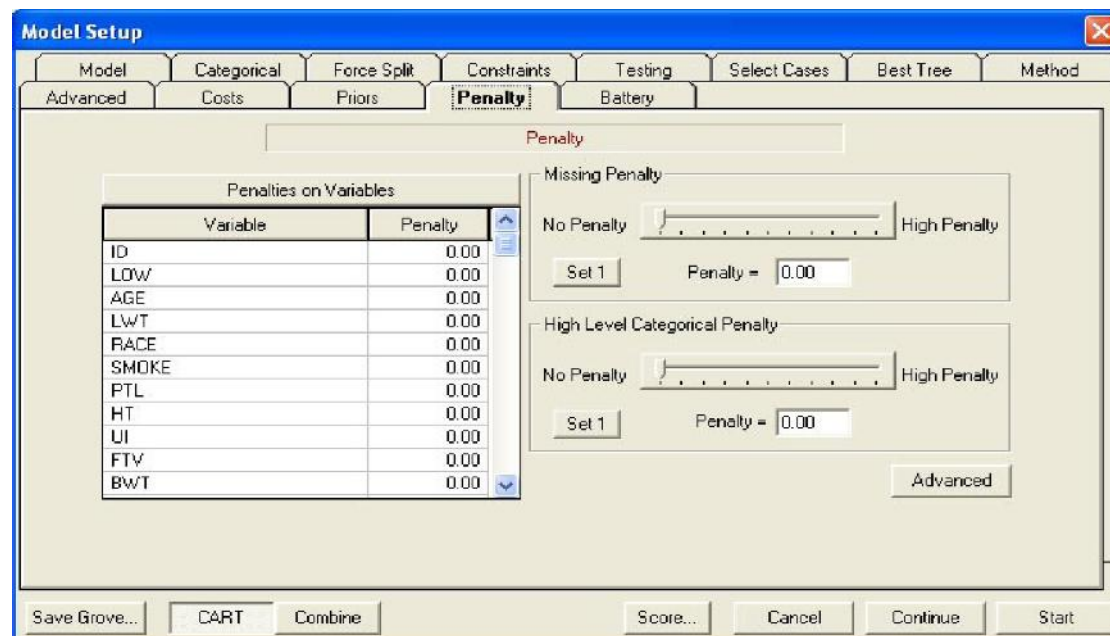
III. Setting Up CART Model

THE PENALTY TAB

Penalties can be imposed on variables to reflect the reluctance to use a variable as a splitter. Of course, the modeler can always exclude a variable; the penalty offers an opportunity to permit a variable into the tree but only under special circumstances.

A penalty will lower a predictor's improvement score, thus making it less likely to be chosen as the primary splitter.

A Snapshot of Penalty Tab



Model Setup

Model | Categorical | Force Split | Constraints | Testing | Select Cases | Best Tree | Method

Advanced | Costs | Priors | **Penalty** | Battery

Penalty

Penalties on Variables

Variable	Penalty
ID	0.00
LOW	0.00
AGE	0.00
LWT	0.00
RACE	0.00
SMOKE	0.00
PTL	0.00
HT	0.00
UI	0.00
FTV	0.00
BWT	0.00

Missing Penalty

No Penalty High Penalty

Set 1 Penalty = 0.00

High Level Categorical Penalty

No Penalty High Penalty

Set 1 Penalty = 0.00

Advanced

Save Grove... | CART | Combine | Score... | Cancel | Continue | Start

THREE CATEGORIES OF PENALTY

- **Missing Value Penalty:**

Predictors are penalized to reflect how frequently they are missing. The penalty is recalculated for every node in the tree.

- **High Level Categorical Penalty:**

Categorical predictors with many levels can distort a tree due to their explosive splitting power. The HLC penalty levels the playing field.

- **Predictor Specific Penalties:**

Each predictor can be assigned a custom penalty. Setting the penalty to one is equivalent to effectively removing that predictor from the predictor list.

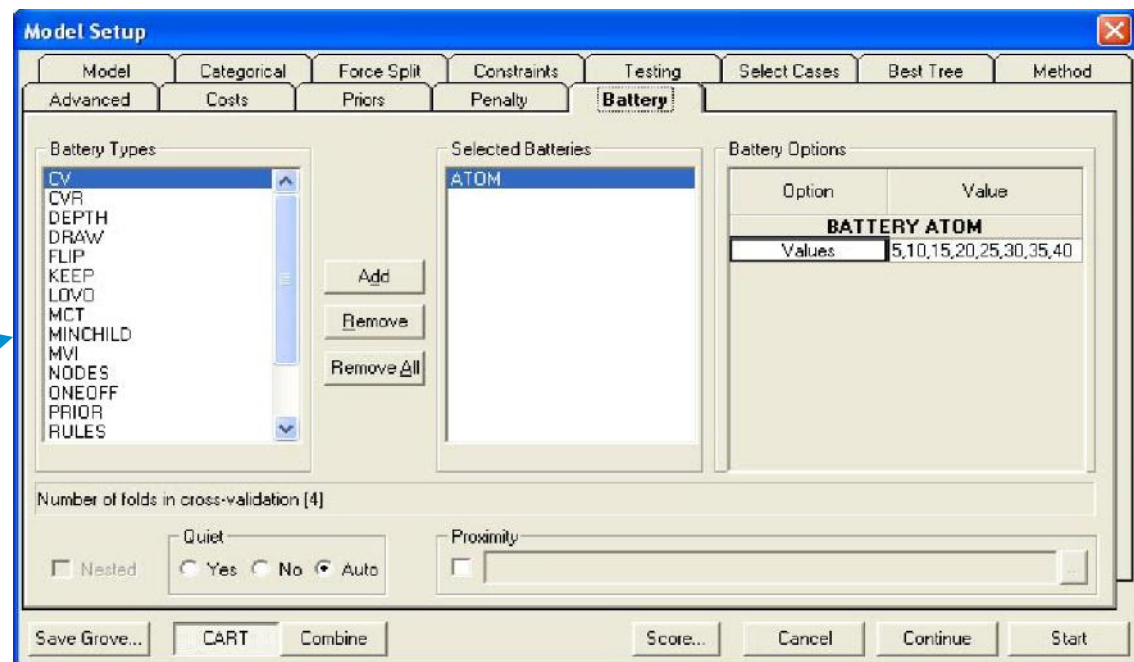
Note: Penalties for missing values (for categorical and continuous predictors) and a high number of levels (for categorical predictors only) can range from "No Penalty" to "High Penalty".

III. Setting Up CART Model

THE BATTERY TAB

The optimal values for many parameters in CART algorithm cannot be determined beforehand and require a trial and error experimental approach. CART batteries are *designed to automate the most frequently occurring modeling situations* requiring multiple collections of CART runs.

A Snapshot of Battery Tab



There are numerous battery types available for use.

III. Setting Up CART Model

THE BATTERY TAB

Battery Type	Function	Illustrative Valid Values / Actions
ATOM	Varies the atom size (the minimum required parent node size)	5,10,15,20,25,30,35,40
CV	Runs Cross Validation	5,10,20,50
CVR	Runs multiple CV using different random number seeds	20 cycles
DEPTH	Specifies the depth limit of the tree	2,3,4,5,6,7,8,9
DRAW	Runs a series of models where the learn sample is repeatedly drawn (without replacement) from “main” learn sample as specified by the Testing tab; the test sample is not altered	70% Learn, 30% Test: Twenty 50% drawings from the Learn partition
FLIP	Generates two runs with the meaning of learn and test samples flipped	Two runs by default: Original & Flip
KEEP	Randomly selects a specified number of variables from the initial list of predictors and repeats the random selection multiple times.	Sampling 10 predictors, thirty times
LOVO Leave One Variable Out	Generates a sequence of runs where each run omits one of the variables on the predictor list one at a time. Assuming K predictors on the initial keep list, the battery produces K models having K-1 predictors each.	Specify a set of K predictors
MCT	Generates a Monte Carlo test on the significance of model performance	Specify a set of K predictors
MINCHILD	Varies the required minimum terminal node size	5,10,15,20,25,30,35,40
MVI	Addresses missing value handling by running a series of five models: MVI_No_P : use regular predictors, missing value indicators, and no missing value penalties No_MVI_No_P : use regular predictors only (default CART model, no MVIs, no penalties) MVI_only : use missing value indicators only (no regular predictors, no penalties) MVI_P : use regular predictors, missing value indicators, and missing value penalties No_MVI_P : use regular predictors and missing value penalties (no MVIs)	Specify a set of K predictors

III. Setting Up CART Model

THE BATTERY TAB

Battery Type	Function	Illustrative Valid Values / Actions
NODES	Varies the limit on the tree size in nodes according to a user-supplied setting	10,20,30,50,100
ONEOFF	Displays results of using one variable at a time to predict the response Note: Better than correlation analysis, tries to identify potential non-linearity as well.	Specify a set of K predictors
PRIOR	Allows priors to be varied within the specified range in user-supplied increments	(0.05, 0.95) to (0.95, 0.05) in increments of 0.05 (19 runs)
RULES	Runs each available splitting method Six Classification Runs for: ▪Gini ▪Symmetric Gini ▪Entropy ▪Class Probability ▪Twoing ▪Ordered Twoing Two Regression Runs for: ▪ Least Squares ▪ Least Absolute Deviation	Specify a set of K predictors
SAMPLE	Investigates the amount of accuracy loss incurred in the course of progressive reduction of the train data size (observation-wise): FIVE runs are produced	Training Data: 100%, 75%, 50%, 25% and 12.5%
SHAVING	Eliminates one or a group of variables based on a specified strategy ▪BOTTOM : Remove the least important variables (up to K runs) ▪TOP : Remove the most important variables (up to K runs) ▪ERROR : Remove the variable with the least contribution based on LOVO battery	Specify a set of K predictors
SUBSAMPLE	Varies the sample size used at each node to determine competitors & surrogates	100, 250, 500, 1000 and 5000
TARGET	Takes each variable from the current predictor list as a target and builds a model to predict this target (classification tree for categorical predictors and regression tree for continuous predictors) using the remaining variables.	Specify a set of K predictors

III. Setting Up CART Model

OTHER FEATURES

■ Unsupervised Learning

- *This does not begin with a target variable. Instead the objective is to find groups of similar records in the data. One can think of unsupervised learning as a form of data compression: we search for a moderate number of representative records to summarize or stand in for the original database.*

■ Force Splits:

- The Model Setup – Force Split tab allows the user to dictate the splitter to be used in the root node (primary splitter), or in either of the two child nodes of the root. Users wanting to impose some modest structure on a tree frequently desire this control. More specific controls also allow the user to specify the split values for both continuous and categorical variables.

■ Constraints:

- By default, all predictors are allowed to be used as primary splitters and as surrogates at all depths and node sizes. The Model Setup – Constraints tab is used to specify at which depths and in which partitions (by size) the predictor, or group of predictors, are not permitted to be used, either as a splitter, a surrogate, or both.

■ Ensemble Models:

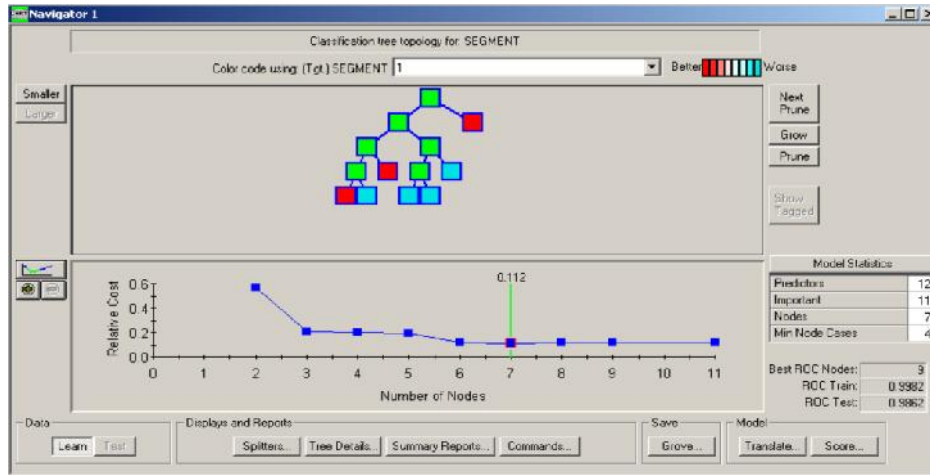
- CART's Combine dialog allows the user to choose from two methods for combining CART trees into a single predictive model. In both **Bootstrap Aggregation (Bagging)** and **Adaptive Resampling and Combining (ARCing)**, a set of trees is generated by resampling with replacement from the original training data.

Bagging : Each new resample is drawn in an identical way (independent samples)

ARCing : The way a new sample is drawn for the next tree depends on the performance of the prior trees. Cases that get misclassified in previous trees receive an increasing probability of selection in the next sample; while cases getting correctly classified in previous trees receive declining weights.

IV. CART Output

NAVIGATOR



Navigator provides an immediate snapshot of the tree's size and depth. By default, the optimal or minimum cost tree is initially displayed.

*To view the **entire tree**, click on the [Tree Details...] button at the bottom of the Navigator.*

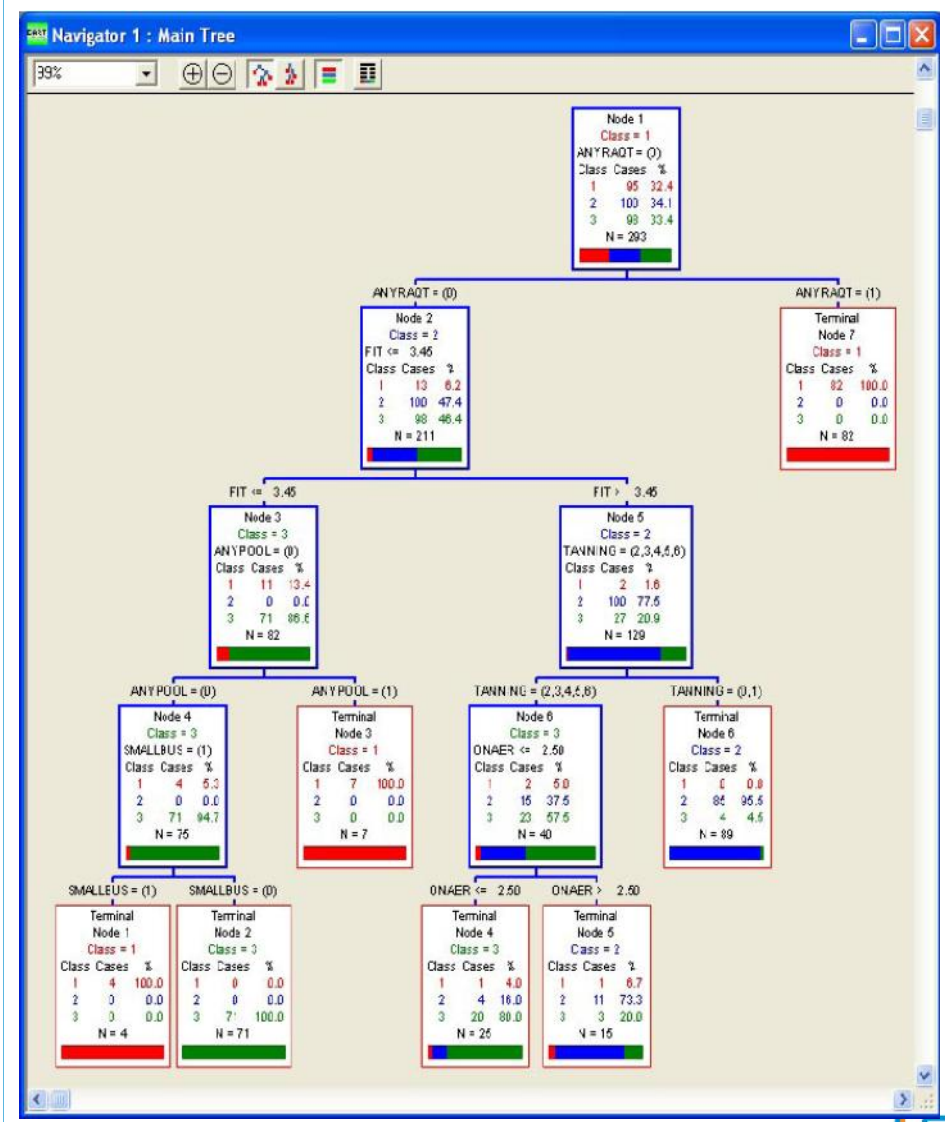
*To get a **SAS code** for the current CART model, click on the [Translate...] button at the bottom of the Navigator.*

*To **score new dataset** using current CART Model, click on the [Score...] button at the bottom of the Navigator.*



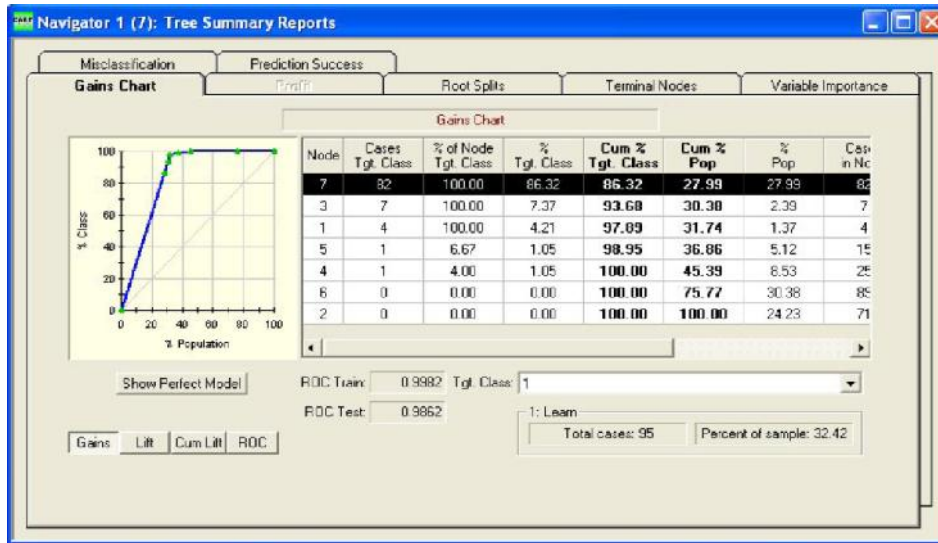
Do remember to save grove for any future reference.

TREE DETAILS



IV. CART Output

GAINS CHART



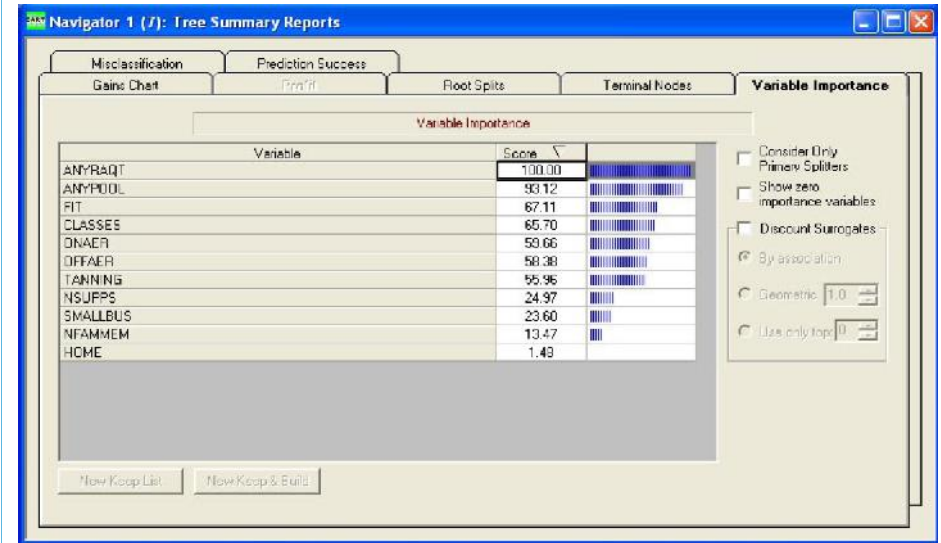
The overall performance of the current tree is summarized in the Summary Reports dialog tabs. To access the reports, click [Summary Reports...] at the bottom of the Navigator window.

Gains Chart: The vertical difference between two lines depicts the **gain from CART model** at each point (over any random model) along the x-axis. The Gains Table can be exported to Excel by a right-mouse click and then choosing [Export...] from pop-up menu.

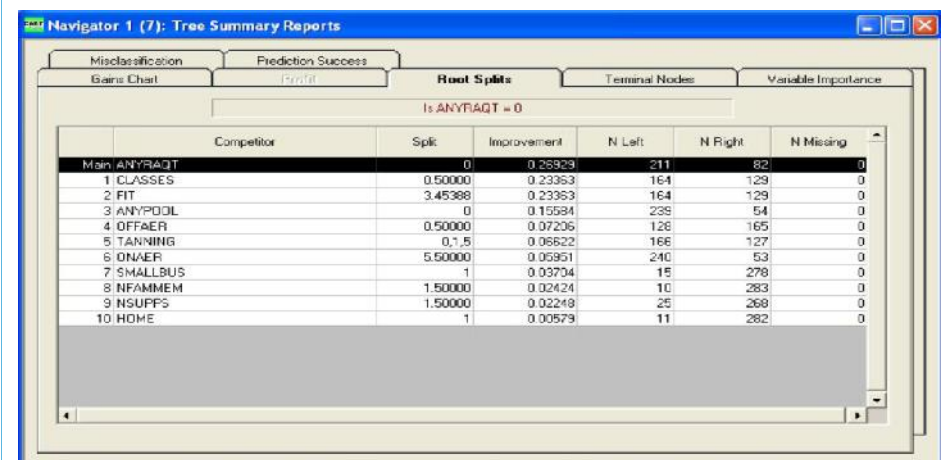
Variable Importance: The scores reflect the contribution each variable makes in classifying or predicting the target variable, with the contribution stemming from both the **variable's role** as a **primary splitter** and as a **surrogate** to other primary splitters.

Root Splits: The report shows the **competing root node splits** in reverse order of improvement.

VARIABLE IMPORTANCE



ROOT SPLITS



IV. CART Output

MISCLASSIFICATION

Navigator 1 (7): Tree Summary Reports

Gain Chart | Profit | Root Splits | Terminal Nodes | Variable Importance

Misclassification | Prediction Success

Misclassification

Learning Sample					Test Sample				
Class	N Cases	N Mis-Classified	Pct Error	Cost	Class	N Cases	N Mis-Classified	Pct Error	Cost
1	95	2	2.11	0.02	1	95	2	2.11	0.02
2	100	4	4.00	0.04	2	100	5	5.00	0.05
3	98	7	7.14	0.07	3	98	15	15.31	0.15

Sort by: Class

Misclassification: The report shows how many cases were incorrectly classified in the overall tree for both learn and test (or cross-validated) samples.

Node Rule: The Terminal node reports (with the exception of the root node) contain a Rules dialog that displays the rules for the selected node and/or sub-tree.

Prediction Success: The Prediction Success table (also known as the confusion matrix) shows whether CART tends to concentrate its misclassifications in specific classes and, if so, where the misclassifications are occurring.

NODE RULES

Navigator 1 (7): Node 5

Competitors and Surrogates | Classification | Rules | Splitter

```

Rules for node 5:
if
{
  ANYRAQT == 0
  &&
  HT > 3.45388
}
{
  node = 5;
  class = 2;
}
  
```

Notation: Classic SQL

Probabilities: None Learn Test Pooled

PREDICTION SUCCESS

Navigator 1 (7): Tree Summary Reports

Gain Chart | Profit | Root Splits | Terminal Nodes | Variable Importance

Misclassification | Prediction Success

Learning Sample Prediction Success Table

Actual Class	Total Cases	Percent Correct	Predicted Class		
			1 N=93	2 N=104	3 N=96
1	95	97.89	93	1	1
2	100	96.00	0	96	4
3	98	92.86	0	7	91
Total	293.00				
Average		95.58			
Overall % Correct		95.56			

Learn Test Class: None

Count: Row % Column %

V. Introduction to MARS

MULTIVARIATE ADAPTIVE REGRESSION SPLINES (MARS)

DESCRIPTION

MARS (Multivariate Adaptive Regression Splines) is a multivariate non-parametric regression procedure which builds flexible regression models by fitting separate Splines (or basis functions) to distinct intervals of the predictor variables.

COMPARISON

CART is also a non-linear, non-parametric technique. However, whereas CART simply finds a cut-point in the predictor variable that classifies the records into different levels of the response variable, MARS finds a “knot” that separates two dynamic relationships between predictor and response.

SOFTWARE

MARS

APPLICATIONS

Creation of basis functions; i.e. non-linear discontinuous transformations of predictors that improve the correlation with the response variable
Regression Models that use basis function transformations to enhance predictive power

VI. Spline Transformation

Basis Function Transformations:

MARS can be used for predictive modeling of continuous outcomes or to model binary metrics by providing a predicted probability of an outcome.

MARS can be used to create spline transformations of variables for use in linear and logistic regression models, hence improving their accuracy.

The core building block of a MARS model is the basis function transformation of a predictor variable X :

$$\text{Basis function} = \text{Max}(0, X-C) \quad \text{where } C \text{ is a constant discovered by the algorithm}$$

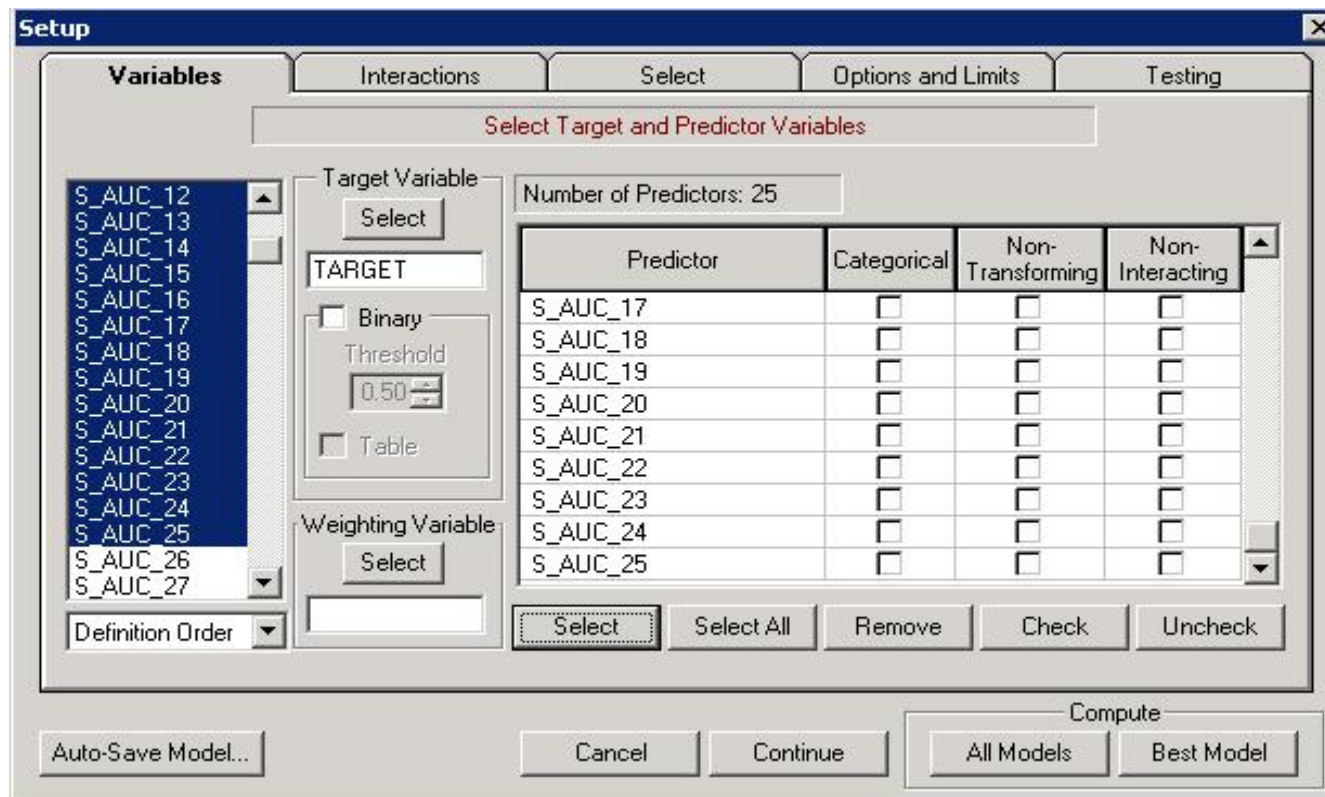
The value of this basis function is equal to 0 for all values of X up to a threshold C and equal to $X-C$ for all values of X greater than C . The Basis function defines a knot (C) where a regression changes slope.

When used in a subsequent linear model, the model equation is a linear combination of the basis functions and their interactions.

VII. Setting Up MARS Model

Building MARS Model:

MARS model can be constructed by simply opening a data file, specifying the target variable and predictors and clicking on “Best Model” in the model setup dialogue.



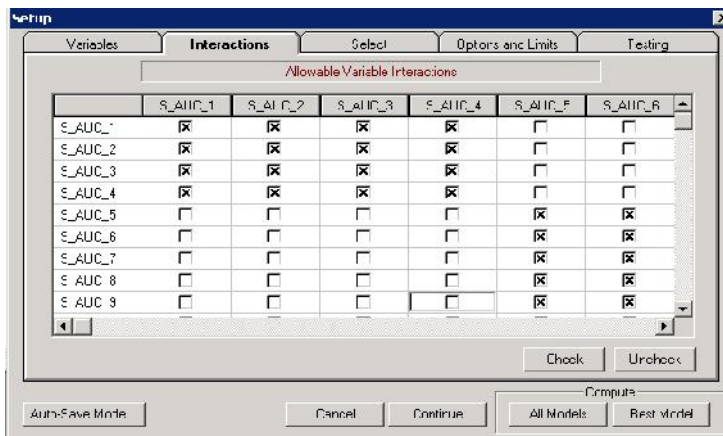
The screenshot shows the 'Setup' dialog box for building a MARS model. The 'Variables' tab is active, showing a list of variables on the left and a table of predictors on the right. The 'Target Variable' is set to 'TARGET'. The 'Number of Predictors' is 25. The 'Weighting Variable' is empty. The 'Definition Order' is set to 'Definition Order'. The 'Compute' section at the bottom has buttons for 'All Models' and 'Best Model'.

Predictor	Categorical	Non-Transforming	Non-Interacting
S_AUC_17	☐	☐	☐
S_AUC_18	☐	☐	☐
S_AUC_19	☐	☐	☐
S_AUC_20	☐	☐	☐
S_AUC_21	☐	☐	☐
S_AUC_22	☐	☐	☐
S_AUC_23	☐	☐	☐
S_AUC_24	☐	☐	☐
S_AUC_25	☐	☐	☐

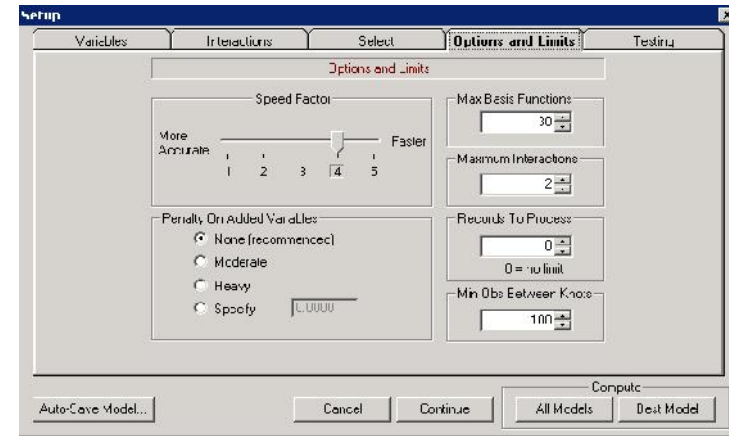
VII. Setting Up MARS Model

Selecting Options:

MARS also allows the user to have more control over the model build. E.g. users can specify the maximum interactions, maximum basis functions, threshold, minimum observations between knots etc.



	S_A1I1C_1	S_A1I1C_2	S_A1I1C_3	S_A1I1C_4	S_A1I1C_5	S_A1I1C_6
S_AUC_1	☒	☒	☒	☒	☐	☐
S_AUC_2	☒	☒	☒	☒	☐	☐
S_AUC_3	☒	☒	☒	☒	☐	☐
S_AUC_4	☒	☒	☒	☒	☐	☐
S_AUC_5	☐	☐	☐	☐	☒	☒
S_AUC_6	☐	☐	☐	☐	☒	☒
S_AUC_7	☐	☐	☐	☐	☒	☒
S_AUC_8	☐	☐	☐	☐	☒	☒
S_AUC_9	☐	☐	☐	☐	☒	☒



Options and Limits

Speed Factor: More Accurate (1) 2 3 4 5 Faster

Max Basis Functions: 30

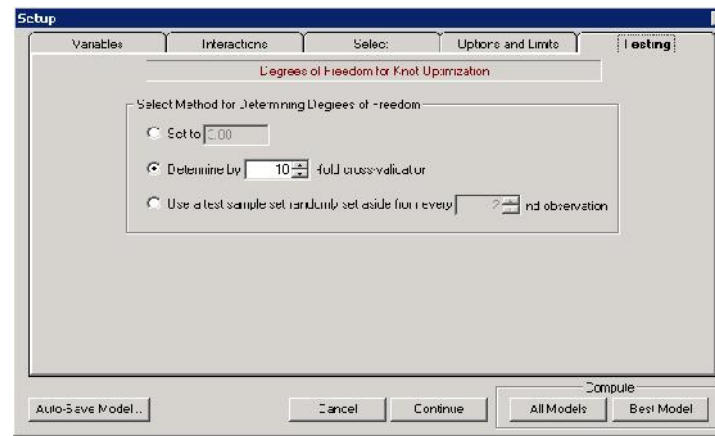
Maximum Interactions: 2

Penalty On Added Variables:

- ☒ None (recommended)
- ☐ Moderate
- ☐ Heavy
- ☐ Spooky

Records To Process: 0 (0 = no limit)

Min Obs Between Knots: 1000



Testing

Degrees of Freedom for Knot Optimization

Select Method for Determining Degrees of Freedom:

- ☐ Set to 0.00
- ☒ Determine by 10 full cross-validation
- ☐ Use a test sample set randomly set aside from every 2nd observation

VII. Setting Up MARS Model

Selecting Options:

Some of the typical controls that users choose to set:

- *Maximum interactions*

Specifies the highest degree of interaction allowed. The default 1 will not allow any interactions, irrespective of what was specified in the interactions tab discussed earlier. A setting of 2 will allow for 2 way interactions and so on. Note that for MARS to explore the interactions properly the maximum allowed Basis functions should be increased in comparison to a no interaction setup

- *Maximum basis functions*

Upper bound on basis functions. Usually at each step 2 Basis functions are added, so if the limit is specified at 15 we will have about 7 forward steps in model development. As a rule of thumb the maximum number of Basis functions should be specified at 2-4 times the number we think is optimal. This in itself might involve a lot of judgment and trial and error! The larger this maximum is set, the longer a MARS run will take.

- *Number of records to process*

This allows us to model with fewer observations. The default 0 implies that the entire dataset will be used

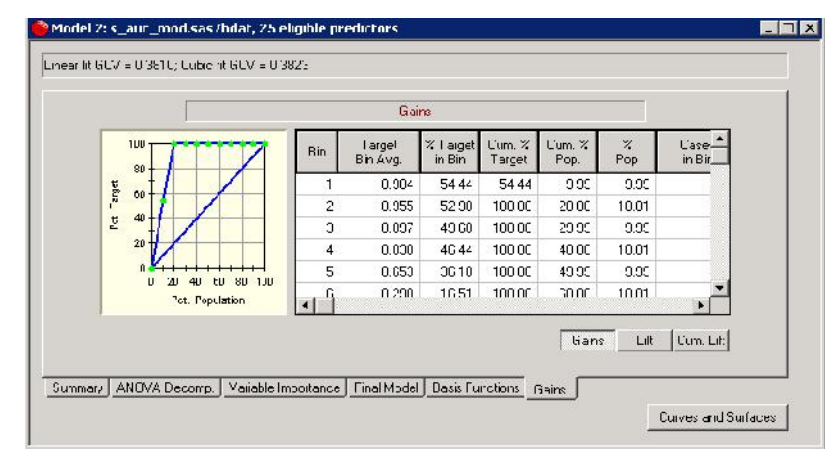
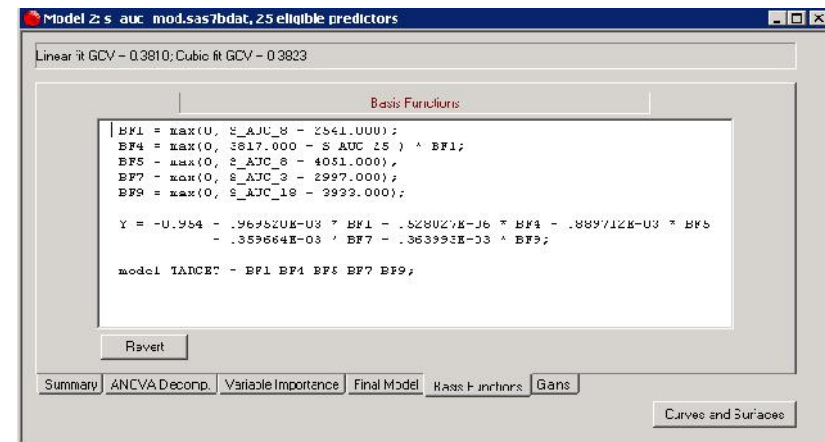
- *Minimum observations between knots*

By default MARS allows a knot to be formed at each and every observation. However it is advisable to make the model a little more general and less locally adaptive. To do this the minimum observations between knots can be set to a value like 100. Again, a lot of judgment is required in making these calls.

VIII. MARS Output

Model Output:

When MARS finds the best model, It outputs the basis functions, model performance, model formula, the variable importance ranks and other information.



VIII. MARS Output

Key Output as Displayed by Model Summary Page:

- **Target variable:** Name, min, max, mean and variance of the target variable
- **Direct variables:** number of variables used to construct basis functions
- **Total variables used:** Total number of variables used to construct basis functions
- **Terms in the model:** Number of coefficients in the MARS model excluding the intercept
- **Effective parameters:** Based on number of terms, number of knots, and degrees of freedom per knot
- **Naïve MSE:** Mean square error from regression equation
- **MARS GCV:** Penalized mean-square error (calculated by dividing the error sum of squares by $N-M$ instead of N , where N is the sample size and M is the number of basis functions).
- **Naïve R^2 :** R^2 value for regression equation using final MARS model
- **Naïve-Adjusted R^2 :** Adjusted R^2 for MARS regression model
- **GCV R-square:** $1 - \text{Final-GCV} / \text{Initial-GCV}$

VIII. MARS Output

Model Assessment:

- On a Naive R^2 criterion, adding more basis functions always reduces the mean square error. So, to protect against over-fitting, **MARS uses a penalty to adjust R^2 and MSE**. The penalty is similar in spirit to AIC (Akaike Information Criterion), but is determined dynamically from the data. The optimal MARS model is the one with the lowest GCV measure.
- Model performance is measured by the Gains Chart/Table and Prediction Success table (if target variable is binary). The Gains Table is constructed by scoring and sorting in descending order the training data, divided into 10 equal deciles, and displaying the average predicted and actual values within each decile. The **Gains Chart** plots the **lift versus the population**. Like a contingency table, the **Prediction Success Table** displays the **numbers of events and non-events correctly and falsely predicted**.
- While CART handles classification problems better than MARS, it can be deficient when it comes to regression. A decision tree with 30 terminal nodes is capable of making only 30 distinct predictions (one per node); thus, all records landing in a node receive exactly the same prediction. MARS is capable of predicting with much higher resolution and accuracy, typically producing **unique scores for every record in a database**.
- **MARS can be used in conjunction with CART**. CART first can be used to extract the most important variables from a very large list of potential predictors. MARS can then **focus on the top variables from the CART model**, resulting in faster MARS analyses and more accurate and robust models.

IX. CART and MARS for Creating Transforms

SOFTWARE	PROCEDURE	USAGE
CART	▪ Build shallow CART tree (2 or 3 levels) ▪ Create binary indicator variable for each terminal node	▪ Indicators represent key non-linear interactions of splitting variables used to create nodes ▪ Could not be discovered by linear modeling search procedures ▪ Can be used as independent variables in subsequent models to extend model discrimination and explanatory power
MARS	▪ Create new variables representing these transformations ▪ Run MARS to discover basis functions with best (non-linear) correlation with response	▪ New variables represent key non-linear spline transformations ▪ Could not be discovered by linear modeling search procedures ▪ Can be used as independent variables in subsequent models to extend model discrimination and explanatory power

Thanks

For queries, contact Varun Aggarwal at Varun.Aggarwal@exlservice.com